```
WHETSTONEBMK|
begin

real x1, x2, x3, x4, x, y, z , t, t1, t2, a, b, c;
array e[1:4];
integer i, j , k, l, n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11;

procedure pa(ep);
   real array ep;
begin
   integer j ;
   j := 0;
   lab:
   ep[1] := (ep[1] + ep[2] + ep[3] - ep[4])  × t;
   ep[2] := (ep[1] + ep[2] - ep[3] + ep[4])  × t;
   ep[3] := (ep[1] - ep[2] + ep[3] + ep[4])  × t;
   ep[4] := (-ep[1] + ep[2] + ep[3] + ep[4]) / t2;
   j := j + 1;
   if j < 6 then
   goto lab;
end procedure pa;

procedure p0;
begin
   e[j] := e[k];
   e[k] := e[l];
   e[l] := e[j];
end procedure p0;

procedure p3(x, y, z );
   value x, y;
   real x, y, z ;
begin
   x := t × (x + y);
   y := t × (x + y);
   z := (x + y)/t2;
end procedure p3;

procedure pout(n, j , k, x1, x2, x3, x4);
   value n, j , k, x1, x2, x3, x4;
   integer n, j , k;
   real x1, x2, x3, x4;
begin
   writetext(30, [n=]);   write(30, format([nnnd]), n);
   writetext(30, [*j=]);  write(30, format([nnnd]), j);
   writetext(30, [*k=]);  write(30, format([nnnd]), k);
   writetext(30, [*x1=]); write(30, format([+d.ddddddddd]), x1);
   writetext(30, [*x2=]); write(30, format([+d.ddddddddd]), x2);
   writetext(30, [*x3=]); write(30, format([+d.ddddddddd]), x3);
   writetext(30, [*x4=]); write(30, format([+d.ddddddddd]), x4);
   writetext(30, [[c]]);
end procedure pout;

open(30);

comment initialise constants;
t  := 0.499975;
t1 := 0.50025;
t2 := 2.0;

comment read i, controlling total weight:
   if i=1 the total weight is one million Whetstone instructions:
   Walgol ran at 2.4KWIPS, so this should take 417 KDF9 CPU seconds;
i := 1;
i  := i × 10;
n1 := 0;
n2 := 12 ×i;
n3 := 14 ×i;
n4 := 345 ×i;
n5 := 0;
n6 := 210 ×i;
n7 := 32 ×i;
n8 := 899 ×i;
n9 := 616 ×i;
n10 := 0;
n11 := 93 ×i;

comment module 1: simple identifiers;
x1 := 1.0;
x2 := x3 := x4 := -1.0;
for i := 1 step 1 until n1 do
begin
   x1 := (x1 + x2 + x3 - x4) ×t;
   x2 := (x1 + x2 - x3 + x4) ×t;
   x3 := (x1 - x2 + x3 + x4) ×t;
   x4 := (-x1 + x2 + x3 + x4) ×t;
end module 1;
pout(n1, n1, n1, x1, x2, x3, x4);

comment module 2: array elements;
e[1] := 1.0;
e[2] := e[3] :=e[4] := -1.0;
for i := 1 step 1 until n2 do
begin
   e[1] := (e[1] + e[2] + e[3] - e[4])  × t;
   e[2] := (e[1] + e[2] - e[3] + e[4])  × t;
   e[3] := (e[1] - e[2] + e[3] + e[4])  × t;
   e[4] := (-e[1] + e[2] + e[3] + e[4]) × t;
end module 2;
pout(n2, n3, n2, e[1], e[2], e[3], e[4]);

comment module 3: array as parameter;
for i := 1 step 1 until n3 do
   pa(e);
pout(n3, n2, n2, e[1], e[2], e[3], e[4]);

comment module 4: conditional jumps;
j := 1;
for i := 1 step 1 until n4 do
begin
   if j = 1 then
      j := 2
   else
      j := 3;
   if j > 2 then
      j := 0
   else
      j := 1;
   if j < 1 then
      j := 1
   else
      j := 0;
end module 4;
pout(n4, j , j , x1, x2, x3, x4);

comment module 5: omitted;

comment module 6: integer arithmetic;
j := 1;
k := 2;
l := 3;
for i := 1 step 1 until n6 do
begin
   j := j × (k - j ) × (l - k);
   k := l × k - (l - j ) × k;
   l := (l - k) × (k + j );
   e[l - 1] := j + k + l;
   e[k - 1] := j × k × l;
end module 6;
pout(n6, j , k, e[1], e[2], e[3], e[4]);

comment module 7: trig. functions;
x := y := 0.5;
for i := 1 step 1 until n7 do
begin
   x := t × arctan(t2 × sin(x) × cos(x) / (cos(x + y) + cos(x - y) - 1.0));
   y := t × arctan(t2 × sin(y) × cos(y) / (cos(x + y) + cos(x - y) - 1.0));
end module 7;
pout(n7, j , k, x, x, y, y);

comment module 8:
procedure calls;
x := y := z := 1.0;
for i := 1 step 1 until n8 do
begin
   p3(x, y, z );
end module 8;
pout(n8, j , k, x, y, z , z );

comment module 9: array references;
j := 1;
k := 2;
l := 3;
e[1] := 1.0;
e[2] := 2.0;
e[3] := 3.0;
for i := 1 step 1 until n9 do
   p0;
pout(n9, j , k, e[1], e[2], e[3], e[4]);

comment module 10: integer arithmetic;
j := 2;
k := 3;
for i := 1 step 1 until n10 do
begin
   j := j + k;
   k := j + k;
   j := k - j ;
   k := k - j - j ;
end module 10;
pout(n10, j , k, x1, x2, x3, x4);

comment module 11: standard functions;
x := 0.75;
for i := 1 step 1 until n11 do
begin
   x := sqrt(exp(ln(x)/t1));
end module 11;
pout(n11, j , k, x, x, x, x);

close(30);

end
```